e-Informatica Software Engineering Journal, Volume 17, Issue 1, 2023, pages: 230104, DOI: 10.37190/e-Inf230104

Story Point Estimation Using Issue Reports With Deep Attention Neural Network

Haithem Kassem*, Khaled Mahar**, Amani A. Saad***
*Multimedia Center, AASTMT, Alex, Egypt
**College of Computing and Information Technology, AASTMT, Alex, Egypt
***College of Engineering and Technology, AASTMT, Alex, Egypt
haithem_k@aast.edu, khmahar@aast.edu, amani.saad@aast.edu

Abstract

Background: Estimating the effort required for software engineering tasks is incredibly tricky, but it is critical for project planning. Issue reports are frequently used in the agile community to describe tasks, and story points are used to estimate task effort.
Aim: This paper proposes a machine learning regression model for estimating the number of story points needed to solve a task. The system can be trained from raw input data to predict outcomes without the need for manual feature engineering.
Method: Hierarchical attention networks are used in the proposed model. It has two levels of attention mechanisms implemented at word and sentence levels. The model gradually constructs a document vector by grouping significant words into sentence vectors and then merging significant sentence vectors to create document vectors. Then, the document vectors are fed into a shallow neural network to predict the story point.
Results: The experiments show that the proposed approach outperforms the state-of-the-art technique Deep-S which uses Recurrent Highway Networks. The proposed model has improved Mean Absolute Error (MAE) by an average of 16.6% and has improved Median Absolute Error ($MdAE$) by an average of 53%.
Conclusion: An empirical evaluation shows that the proposed approach outperforms the previous work.

Keywords: story points, deep learning, glove, hierarchical attention networks, agile, planning poker

1. Introduction

The primary goal of all software project managers is to complete the project on time and within the budget that has been established. Since the release of the agile manifesto [1], many companies have chosen to use agile approaches to guide software development. Estimating effort is critical for successful agile project management. To avoid inefficient resource allocation, accurate estimates are required [2, 3]. The story points [4, 5] are a popular method for estimating task effort. In the context of agile development, story points are typically assigned through organized group meetings known as Planning Poker sessions [6]. These meetings heavily rely on human judgment: the better the developers understand the job, the more accurate their estimates will be. Human judgment, on the other hand, is sensitive to a range of constraints. Humans are positive by nature, and

Submitted: 27 Mar. 2022; Revised: 21 Nov. 2022; Accepted: 22 Nov. 2022; Available online: 20 Jan. 2023

this bias is amplified in group interactions [7–9]. Furthermore, the presence of a project manager, other senior developers, or dominant personalities in the meeting has been shown to affect developer estimation [10].

The use of machine learning regessors has three advantages. To begin with, the regressors have a thorough understanding of the project that dates back to its beginnings, and it based its predictions on all past issues in the issue tracking system. Second, because the regressors' estimations can be tracked back to the regressor's characteristics, it is not influenced or coerced by others. Third, the estimation is repeatable and predictable: the system never grows bored of producing the same results over and over again.

We introduce a prediction model that helps teams by providing a story-point estimate for a certain user story. The model uses the team's previous story point assessments to forecast the complexity of new issues. The team's existing estimation techniques will be used in conjunction with (rather than in place of) this prediction system. It could also be used as a decision-making tool and help with estimating. This is similar to the notion of combination-based effort estimating [11, 12]. Estimates are generated from various sources, such as a combination of expert and formal model-based estimates.

The suggested model automatically learns semantic features that represent the meaning of user stories or issue reports, removing the need for users to develop and extract features manually. Feature engineering is often done by domain specialists who use their in-depth understanding of the data to develop features that machine learning algorithms may exploit. Our model is a full end-to-end system that estimates story points by passing raw data signals (i.e., words) from input nodes to the final output node. The use of hierarchical attention networks (HAN) for story point prediction is a fundamental innovation in our method.

An empirical evaluation was conducted to answer the following research questions:

RQ1. Does the use of Hierarchical Attention Networks provide more accurate story point estimates than Recurrent Highway Nets?

RQ2. Does the use of BERT provide more accurate story point estimates than using HAN?

The remainder of the paper is organized as follows: Section **??** provides context for Story Points, Planning Poker, Deep learning, and Hierarchical Attention Network. Section 3 presents related works, while Section 4 focuses on the design of the proposed model. Section 5 discusses the proposed model evaluation , Section 6 comparing with the state of art, Section 7 shows future work, and finally Section 8 presents the conclusion.

2. Background

2.1. Story points

Story points are a unit of measure for expressing the overall size of a user story,feature or another piece of work [4]. The number of story points is an indication of how difficult a specific task is for the development team, rather than measuring the quantity of work required to achieve it. As a first stage, the team normally decides on the number of story points that a baseline activity deserves. After that, estimating effort is dependent on comparison to that baseline. The Fibonacci sequence (i.e., 1, 2, 3, 5, 8, 13, 21, 34, 55, …) is

commonly deviated from when assigning story points. The uncertainty that comes with estimating complex tasks in real-world software is shown in this series.

2.2. Planning pokers

The majority of software projects rely completely on human judgment to estimate effort [13]. The most prevalent effort estimation approaches based on human judgment are those based on group estimation. When it comes to estimating story points, Planning Poker [13] is the most commonly used method. To perform Planning Poker, the customer must first communicate an issue that they would like to get handled. The developers then gather for a poker game in which each player selects a card with the desired story points for each issue to be estimated, and then all the cards are revealed at the same time. The developer that provides the lowest and highest estimate must justify their choice, thus eventually triggering further discussion which is followed by another group estimation. The process continues until the team agrees upon a consensus estimate.

2.3. Deep learning

Deep learning technology (DL) has shown impressive results in a variety of fields, including machine vision [14], speech recognition [15], and text classification [16]. Researchers can divide deep learning research on text classification into two steps: The first step is to learn word vector representations through neural language models [17], and the second step is to perform classification composition over the learned word vectors.

Deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [18] are commonly used in text classification. Recently, several text classification methods based on CNNs or RNNs have been proposed [19, 20]. The CNN learns local responses using temporal or spatial data, but cannot learn sequential correlations. RNNs, in contrast, are designed to do sequential modeling, but cannot extract features in a parallel manner.

2.4. Hierarchical attention network

Yang et al. [21] developed Hierarchical Attention Network (HANs), a DL document classification model based on RNNs. They are made up of hierarchies, with the lower hierarchies' outputs becoming the upper hierarchies' inputs. This is based on the intuition that documents are made up of meaningful sentences, which are made up of meaningful word sequences. Each HAN hierarchy is made up of a bidirectional dynamic Long short-term memory (LSTM) or gated recurrent unit (GRU) with attention mechanisms. When processing words/sentences, directionality is required so that the network can account for the prior and subsequent context. The attention mechanism is added to enable the network to put extra focus on the LSTM/GRU outputs associated with the words and lines that are most indicative of a particular class. LSTMs/GRUs are used because they allow the network to selectively process input information based on how relevant it is to the classification tasks; similarly, the attention mechanism is added to enable the network to put extra focus on the LSTM/GRU outputs associated with the words and lines that are most indicative of a particular class. Hierarchical attention networks were used to

learn semantic features that automatically convey the true meaning of user stories and predict the estimated story point.We'll go over the details of each component later in this paper.

3. Related work

Methods for estimating software work can be divided into three categories: expert-based, model-based, and hybrid techniques. Expert-based methods, which rely on human understanding to create estimates, are the most widely used technique [22, 23]. Expert-based estimation necessitates the presence of experts at all times when an estimate is required. Model-based approaches draw on data from previous projects, but they differ in terms of how they construct customized models. A fixed model in which elements and variables are fixed is the well-known construction cost (CO-COMO) model [24]. Their relationship has already been established. These estimation models were built using data from a range of past studies. As a result, they are usually only effective for the type of project that was used to develop the model. Regression (e.g., [25, 26]), neural networks (e.g., [27, 28]), fuzzy logic (e.g., [29]), Bayesian belief networks (e.g., [30]), analogy-based (e.g., [31, 32]), and multi-objective evolutionary approaches are all used in the customized model construction process (e.g., [33]). However, no single strategy is expected to perform well across all project types [34–36]. As a result, some recent research [37] suggests integrating estimates from several estimators. Papers [38, 39], which are similar to the ideas in this paper, hybrid techniques integrate expert judgments with available data.

While the majority of existing research focuses on estimating a project as a whole, less attention is paid to developing models for agile projects in particular. Different planning and estimation methodologies are required for today's agile, dynamic, and incremental projects [40]. Machine learning techniques are being used in recent approaches to assist in estimating effort for agile projects. The study recently provided an approach for extracting TF-IDF features from the problem description to construct a model for story point estimations, which was published in [41]. The retrieved features are then subjected to the uniform selection process and input into regressors such as SVM.

In addition, Cosmic Function Points (CFP) [42] estimate the effort required to finish an agile project [43]. Abrahamsson [44] created a regression model and neural network-based effort prediction model for the creation of iterative software. Unlike standard effort estimate models, this model is developed after each iteration (rather than at the end of the design phase) to estimate the effort for the next iteration.

The authors of [45] developed a Bayesian network model for estimating effort in agile Extreme Programming software projects. Their model, on the other hand, is based on several criteria (such as process effectiveness and improvement) that necessitate a significant amount of learning and fine-tuning. Bayesian networks are frequently used in [46] to model dependencies between multiple aspects in Scrum-based software development projects to identify difficulties (e.g., sprint progress and sprint planning quality affect product quality).

Choetkiertikul [47] focuses on estimating issues with story points, which is a substantial improvement over earlier work, by applying deep learning techniques to automatically learn semantic features that reflect the underlying meaning of issue descriptions. The previous study has been done in projecting the elapsed time for correcting a bug or the danger of addressing an issue with a pause (see [48–51]).

The proposed model uses pre-trained embedding vectors and transfer learning with GloVe to save training time, which is the key difference from [47]. Word-to-vector (Word2Vec) and global vector (GloVe) are two recent techniques that are well recognized for producing vector representations [52, 53]. Pennington et al. [54]. demonstrated that GloVe outperforms Word2Vec since Word2Vec has a low vector dimensionality and cannot incorporate all of the corpus data. The GloVe, in comparison, has both local and worldwide information about the words that have appeared. GloVe algorithm uses the statistics of word-word co-occurrences in a corpus and is used for similarity and entity identification [55].

Choetkiertikul [47] is the first model providing end-to-end trainable from raw input data to prediction outcomes without any manual feature engineering and has outperformed previous work [41, 43–46]. The proposed model is aiming to use deep learning and make use of the hierarchical attention mechanism, this model has the ability to detect important words and sentences. The Hierarchical Attention Network (HAN) was implemented with the goal of capturing two fundamental ideas about document organization. First, because documents are hierarchical (words make sentences, sentences form a document), we generate a document representation by first creating sentence representations and then aggregating them into a document representation. Second, different words and sentences in a document are found to have varying levels of information. The model constructs a document vector progressively by aggregating important words into sentence vectors and then aggregating important sentence vectors to document vectors.

4. The proposed model

The general goal of our research is to create a prediction system that takes the title and description of an issue as input and generates the estimated story-point. The proposed model introduces the use of hierarchical attention networks (HAN). An embedding layer, attention layers, and encoders are all components of the HAN model, which together help the model understand the textual features. The extraction of relevant context is the responsibility of the encoders. The attention layers evaluate how important a sequence of tokens is with reference to the document. The HAN essentially consists of "hierarchies," where the outputs of the lower hierarchies serve as the inputs for the upper hierarchies. We first break down a document into sentences before feeding it into the HAN. Each sentence is encoded into a vector representation using a word encoder (a bidirectional GRU) and a word attention mechanism. These sentence representations are passed through a sentence encoder with a sentence attention mechanism resulting in a document vector representation. A fully connected layer with the appropriate activation function receives this final representation and uses it to make predictions. The term "hierarchical" refers to a document's "semantic hierarchy." The same algorithms are used twice, once at the word level and once at the sentence level. The model gradually builds a document vector by grouping significant words into sentence vectors and then merging important sentence vectors to create document vectors. The document vectors are then fed into a shallow neural network to predict the story point. The proposed model is made up of five layers, as shown in Figure 1, and is explained briefly as follows:

1. Input layer: Accepts a document that is made up of sentences, each of which is made up of a series of word IDs that represent user stories or issues that describe what has to be produced in the software project. Assume that a document includes L sentences s_i and each sentence contains T_i words. w_{i_t} with $t \in [1, T]$ represents the words in the i^{th} sentence.

2. Embedding layer: Each word in each sentence is individually embedded, resulting in Sequences of word vectors, one for each sentence. It does this by converting input text into dense word vectors that encode both the meaning and context of the text. Word Representation using Global Vectors Each word's vector representation was obtained using GloVe [56]. GloVe is an unsupervised learning technique for obtaining word vector representations.

3. Encoding layer: We have a sequence of word vectors from the previous layer, and this layer seeks to compute a sentence matrix from which we can construct a document matrix. The sentence matrix is made up of rows, each representing the meaning of a single token in the phrase. A Bidirectional RNN is used to implement this layer. The vector of each token is divided into two portions, one computed with a forward pass and the other with a backward pass. To get the entire vector, we just join the two. There are two encoders in this layer:

Sentence Encoder: Converts sequence of word vectors to sentence matrix.

Document Encoder: Converts sequence of sentence vectors to document matrix.

Given a sentence with words $w_{it}, t \in [0, T]$, we first embed the words to vectors through an embedding matrix $w_e, x_{ij} = w_e w_{ij}$.

We obtain word annotations using bidirectional GRU by combining input from both directions and adding contextual information to the annotation.. The bidirectional GRU contains the forward GRU $\overrightarrow{f}$ which reads the sentence s_i from w_{i1} to w_{iT} and a backward GRU $\overleftarrow{f}$ which reads from w_{iT} to w_{i1}.

$$X_{it} = W_e w_{it}, t \in [1, T]. \tag{1}$$

$$\overrightarrow{h_i t} = \overrightarrow{GRU} X_{it}, t \in [1, T]. \tag{2}$$

$$\overleftarrow{h_i t} = \overleftarrow{GRU} X_{it}, t \in [T, 1]. \tag{3}$$

Document Encoder in a similar manner, given the sentence vectors s_i, we can obtain a document vector. To encode the sentences, we use a bidirectional GRU:

$$\overrightarrow{h_i} = \overrightarrow{GRU}(s_i), i \in [1, L]. \tag{4}$$

$$\overleftarrow{h_i} = \overleftarrow{GRU}(s_i), i \in [L, 1]. \tag{5}$$

4. Attention layer: Our goal in this layer is to reduce the Sentence matrix from the previous layer to a single vector that the feed-forward network may use for prediction. This layer's job is to determine the words that are most important to a user story's meaning. The following equations were utilized in this layer [57] .

$$e_t = \tanh(Uc + Wh_t + b) \tag{6}$$

$$\alpha_t = \text{softmax}(e_t) \tag{7}$$

$$o = \sum \alpha_t h_t \tag{8}$$

c is the vector obtained by applying max poling to the matrix obtained from GRU. U is a new weight.

Output layer: We use a feedforward neural network with a linear activation function as the final regressor to construct a story-point estimate. The following is a definition for

this function:

$$y = a_0 + \sum_{i=1}^{n} a_i x_i \tag{9}$$

where y is the output story point, x_i is an input signal from the previous layer, a_i is the trained coefficient (weight), and n is the size of the embedding dimension.

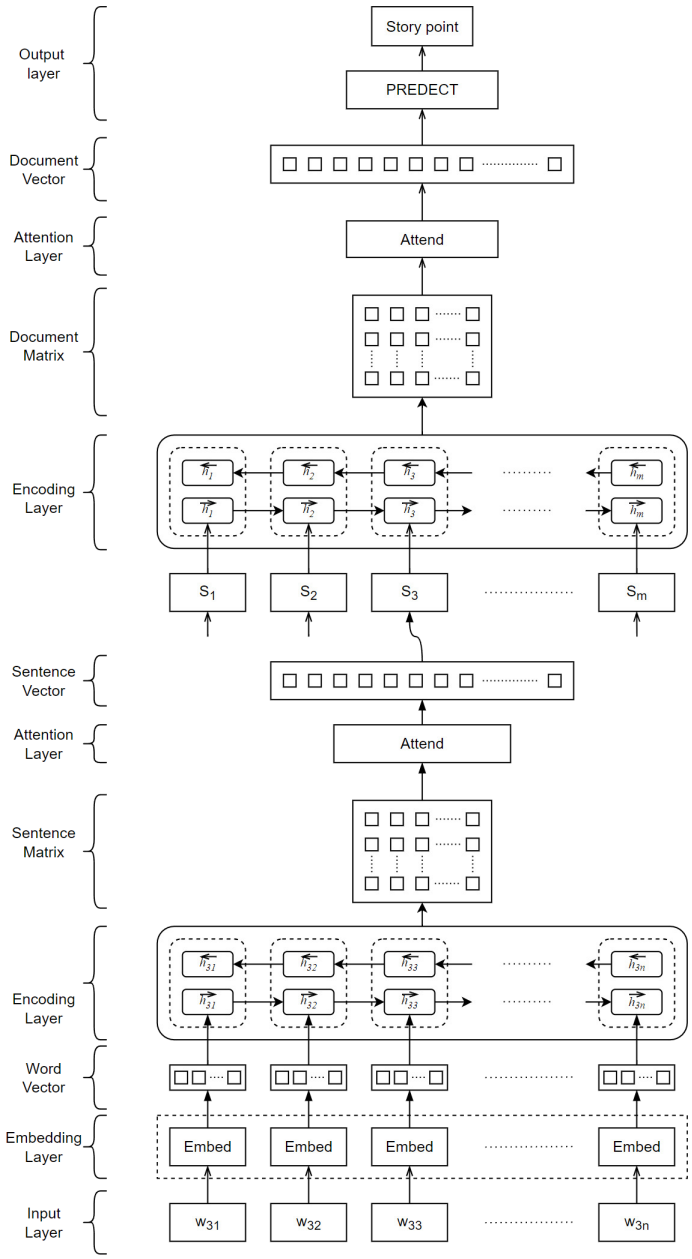

Figure 1. The proposed model

5. The proposed approach evaluation

Our data set [47] includes 23,313 issues from 16 different projects, including Apache Mesos (ME), Apache Usergrid (UG), Appcelerator Studio (AS), Aptana Studio (AP), Titanium SDK/CLI (TI), DuraCloud (DC), Bamboo (BB), Clover (CV), JIRA Software (JI), Moodle (MD), Data Management (DM), Mule (MU), Mule Studio (MS), Spring XD (XD), Talend Data Quality (TE) as shown in Table 1. The data set is divided into three parts: 60% for training, 20% for validation, and the remaining 20% for testing. (Our dataset & code are available at https://doi.org/10.5281/zenodo.7341235). In planning poker, the story points are typically ordered in a Fibonacci sequence, such as 1, 2, 3, 5, 8, 13, 21, and so on. To divide documents into sentences and tokenize each sentence, we used Natural Language Toolkit (NLTK) [58]. The suggested technique employs a Transfer Learning model called pre-trained word embedding. The basic concept is to leverage publicly available embeddings that have been trained on large datasets. Instead of randomly initializing our neural network weights, we use these previously trained integrations as initialization weights. This speeds up training and improves the performance of NLP models. The most widely used method for obtaining word embedding from text corpora is GloVe [57]. It offers pre-trained embedding based on massive text corpora. GloVe allows for different sizes of embedding. The experiment was conducted with a fifty-embedding size. When evaluating the accuracy of an effort estimating model, a variety of metrics are used. The majority of them (i.e., $|ActualSP - EstimatedSP|$) are based on the Absolute Error, where *AcutalSP* denotes the actual story points awarded to a problem, and *EstimatedSP* denotes the result of estimation. Prediction at level [59], i.e., Pred(l), and Mean of Magnitude of Relative Error (MRE) or Mean Percentage Error have also been employed in effort estimate. However, several investigations [59–62] have discovered that these metrics have a proclivity for underestimation and are not reliable. Consequently, the Mean Absolute Error (*MAE*) and Median Absolute Error (*MdAE*) have been recommended to compare effort estimation performance [63, 64] models. The term *MAE* is defined as

$$MAE = \frac{1}{N}\sum_{i=1}^{n} ActualSP_i - EstimatedSP_i \tag{10}$$

where N is the number of issues used for evaluating the performance (i.e., test set), $ActualSP_i$ is the actual story point, and $EstimatedSP_i$ is the estimated story point, for the issue i. We also report the Median Absolute Error since it is more robust to large outliers. *MdAE* is defined as

$$MdAE = \text{Median}\{|ActualSP_i - EstimatedSP_i|\} \tag{11}$$

where $1 \leq i \leq N$.

5.1. Results analysis and discussion

To compare the proposed regressor with the state of the art, we can refer to the work by Choetkiertikul [47]. They use deep learning approaches to automatically learn semantic characteristics that reflect the underlying meaning of issue descriptions to estimate issues using story points, which is a significant advance over previous work. To reduce the risk of external validity, we examined 23,313 issues across sixteen open source projects, each with its size as shown in Figure 2, complexity, development team, and community. Table 2

Table 1. Descriptive statistics of story point dataset

Repo.	Project	Abb.	# Issues	Min SP	Max SP	Mean SP	Median SP	Mode SP	Var SP	Std SP	Mean TD length	LOC
Apache	Mesos	ME	1680	1	40	3.09	3	3	5.87	2.42	181.12	247,542+
	Usergrid	UG	482	1	8	2.85	3	3	1.97	1.40	108.60	639,110+
Appcelerator	Appcelerator Studio	AS	2919	1	40	5.64	5	5	11.07	3.33	124.61	2,941,856#
	Aptana Studio	AP	829	1	40	8.02	8	8	35.46	5.95	124.61	6,536,521+
	Titanium SDK/CLI	TI	2251	1	34	6.32	5	5	25.97	5.10	205.90	882,986+
DuraSpace	DuraCloud	DC	666	1	16	2.13	1	1	4.12	2.03	70.91	88,978+
Atlassian	Bamboo	BB	521	1	20	2.42	2	1	4.60	2.14	133.28	6,230,465#
	Clover	CV	384	1	40	4.59	2	1	42.95	6.55	124.48	890,020#
	JIRA Software	JI	352	1	20	4.43	3	5	12.35	3.51	114.57	7,070,022#
Moodle	Moodle	MD	1166	1	100	15.54	8	5	468.53	21.65	88.86	2,976,645+
Lsstcorp	Data Management	DM	4667	1	100	9.57	4	1	275.71	16.61	69.41	125,651*
Mulesoft	Mule	MU	889	1	21	5.08	5	5	12.24	3.50	81.16	589,212+
	Mule Studio	MS	732	1	34	6.40	5	5	29.01	5.39	70.99	16,140,452#
Spring	Spring XD	XD	3526	1	40	3.70	3	1	10.42	3.23	78.47	107,916+
Talendforge	Talend Data Quality	TD	1381	1	40	5.92	5	8	26.96	5.19	104.86	1,753,463#
	Talend ESB	TE	868	1	13	2.16	2	1	2.24	1.50	128.97	18,571,052#
	Total		23,313									

shows *MAE* and *MdAE*, achieved from hierarchical attention networks (HAN) against Deep-SE using Recurrent Highway Networks for deep representation of issue reports [47], the proposed model improved *MAE* between 0.7 to 28 percent over Deep-SE and Improved *MdAE* between 18 to 68 percent over Deep-SE . Regardless of the size of the data, the improvement is noticeable. The proposed approach surpasses the previous best baseline methods by 16.5 percent and 19.4 percent for small projects like Apache Usergrid and Clover, respectively. This observation holds true across a variety of larger projects. As seen in Table 1, HAN is the best technique, continuously outperforming Deep-SE across all sixteen projects. RQ1 is answered by this finding.

To compare the performance of two estimating models, we used the Wilcoxon Signed Rank Test [65] to determine the statistical significance of the mean absolute errors obtained by the two models. Because it makes no assumptions about underlying data distributions, the Wilcoxon test is a robustness test. In order to evaluate if there were a good effect of the proposed model for estimating the effort needed for a story point, Wilcoxon signed-rank tests revealed a statistically positive change in effort estimation, $z = -3.517, p = 0.001$ with a medium effect size ($d = 0.6$), Cohen's d effect sizes [66] were calculated. Effect sizes of 0.2 were regarded as small, 0.5 as a medium, and 0.8 as large. So the HAN has medium effect size on *MAE*.

5.2. Threats to validity

We attempted to reduce the validity challenges by using real-world data from issues reported in large open-source projects. These issue reports' titles and descriptions and the actual story points assigned to them were gathered. We are aware that those story points were calculated by human teams, which means they may contain biases and, in some cases, be inaccurate. Datasets of various sizes were used in our study. Additionally, in order to reduce conclusion instability we carefully adhered to current best practices when evaluating effort estimation models. To mitigate threats to external validity, we examined 23,313 issues from sixteen open source projects that differ greatly in size, complexity, developer team, and community. We acknowledge, however, that our dataset would not be representative of

Table 2. Comparison between the proposed model and Deep-SE

Project	Method	*MAE*	*MdAE*
Apache Mesos	Deep-SE	1.02	0.73
	HAN	**0.93**	**0.39**
Apache Usergrid	Deep-SE	1.03	0.80
	HAN	**0.84**	**0.47**
Appcelerator Studio	Deep-SE	1.36	0.56
	HAN	**1.35**	**0.54**
Aptana Studio	Deep-SE	2.71	2.52
	HAN	**2.63**	**1.13**
Titanum	Deep-SE	1.97	1.34
	HAN	**1.70**	**0.54**
DuraCloud	Deep-SE	0.68	0.53
	HAN	**0.6**	**0.12**
Bamboo	Deep-SE	0.74	0.61
	HAN	**0.67**	**0.22**
JIRA Software	Deep-SE	1.38	1.09
	HAN	**1.27**	**0.43**
Moodle	Deep-SE	5.97	4.93
	HAN	**5.66**	**1.56**
Data Management	Deep-SE	3.77	2.22
	HAN	**3.63**	**1.03**
Mule	Deep-SE	2.18	1.96
	HAN	**1.86**	**0.81**
Mule Studio	Deep-SE	3.23	1.99
	HAN	**2.56**	**1.39**
Spring XD	Deep-SE	1.63	1.31
	HAN	**1.20**	**0.51**
Talend Data Quality	Deep-SE	2.97	2.92
	HAN	**2.49**	**1.14**
Talend	Deep-SE	0.64	0.59
	HAN	**0.60**	**0.25**
Clover	Deep-SE	2.11	0.8
	HAN	**1.81**	**0.54**

Table 3. Comparison between the proposed model and BERT

Project	Method	*MAE*
Apache Mesos	HAN	0.93
	BERT	3.39
Apache Usergrid	HAN	0.84
	BERT	3.24
Appcelerator Studio	HAN	1.35
	BERT	2.50
Aptana Studio	HAN	2.63
	BERT	4.18
Titanum	HAN	1.7
	BERT	3.49
DuraCloud	HAN	0.49
	BERT	3.79
Bamboo	HAN	0.67
	BERT	2.76
JIRA Software	HAN	1.27
	BERT	3.13
Moodle	HAN	5.66
	BERT	11.99
Data Management	HAN	3.63
	BERT	7.78
Mule	HAN	1.86
	BERT	3.51
Mule Studio	HAN	2.56
	BERT	3.51
Spring XD	HAN	1.2
	BERT	3.16
Talend Data Quality	HAN	2.49
	BERT	4.04
Talend	HAN	0.60
	BERT	3.42
Clover	HAN	1.81
	BERT	3.87

all types of software projects, particularly in commercial settings (despite the fact that open-source and commercial projects are similar in many ways). The nature of contributors, developers, and project stakeholders is one of the key differences between open-source and commercial projects that may influence story point estimation. More research is required for commercial agile projects.

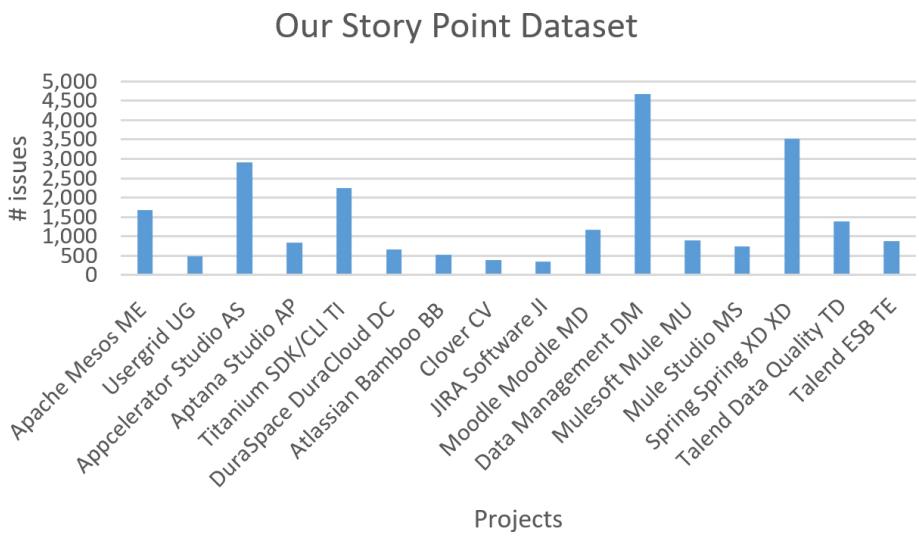

Figure 2. Story point dataset

6. Comparing with the state of art

The Bidirectional Encoder Representations from Transformers (BERT) is a novel approach and is regarded as the cutting edge of pre-trained language representation [67]. BERT models are regarded as contextualized or dynamic models, and they have produced noticeably better results in a number of NLP tasks [68–70], including sentiment classification, calculating the semantic similarity of texts, and identifying tasks involving textual linking. The authors of [52] proposed the use of Bert for effort estimation and their experiments were conducted on the same data set. When comparing our experimental results to [52], the experimental results presented in Table 3 showed that HAN models achieved significantly higher results than the BERT model. RQ2 is answered by this finding. We conclude that a model's performance is dependent on the task and the data, so these factors should be considered before choosing a model rather than just going with the most widely used model.

7. Future work

Future work will involve comparing the results of our model to other pre-trained language models such as GPT [70] and XLNet [71]. These models have been shown to be state of the art in a variety of tasks such as question answering, named entity recognition, and natural language inference. It is also planned to test the proposed model on other Agile data sets.

8. Conclusion

The key novelty of the proposed model is using pre-trained embedding vectors and transfer learning with GloVe to reduce training time instead of creating a new embedding vector. Introducing the use of The Hierarchical Attention Network. The Hierarchical Attention Network (HAN) was created to capture two key concepts in document organization. To begin, we construct a document representation by first creating sentence representations and then aggregating them into a document representation because documents are hierarchical (words make sentences, sentences make a document). Second, different degrees of information are

discovered in different words and phrases in a document. The approach builds a document vector by first aggregating key words into sentence vectors, then aggregating important sentence vectors into document vectors. This process has a significant effect on story point prediction. The results of our experiments show that the proposed model improved *MAE* by 0.7 to 28 percent compared to Deep learning model for Story Point Estimation (Deep-SE) and *MdAE* by 18 to 68 percent compared to Deep-SE. The proposed approach regularly outperforms earlier work. The model can better locate and extract critical information.

References

[1] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham et al., *Manifesto for agile software development*, 2001.

[2] L.C. Briand, "On the many ways software engineering can benefit from knowledge engineering," in *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, 2002, pp. 3–6.

[3] J.W. Paulson, G. Succi, and A. Eberlein, "An empirical study of open-source and closed-source software products," *IEEE Transactions on Software Engineering*, Vol. 30, No. 4, 2004, pp. 246–256.

[4] M. Cohn, "Agile estimating and planning Pearson education," 2006.

[5] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2016, pp. 1–10.

[6] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," *Hawthorn Woods: Renaissance Software Consulting*, Vol. 3, 2002, pp. 22–23.

[7] R. Brown and S. Pehrson, *Group processes: Dynamics within and between groups*. John Wiley & Sons, 2019.

[8] A.R. Lindesmith, A. Strauss, and N.K. Denzin, *Social psychology*. Sage, 1999.

[9] S. Nolen-Hoeksema, B. Fredrickson, G.R. Loftus, and C. Lutz, *Introduction to psychology*. Cengage Learning Washington, 2014.

[10] J. Aranda and S. Easterbrook, "Anchoring and adjustment in software estimation," in *Proceedings of the 10th European Software Engineering Conference Held Jointly With 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2005, pp. 346–355.

[11] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, 2006, pp. 33–53.

[12] K. Moharreri, A.V. Sapre, J. Ramanathan, and R. Ramnath, "Cost-effective supervised learning models for software effort estimation in agile environments," in *40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2. IEEE, 2016, pp. 135–140.

[13] K. Moløkken-Østvold, N.C. Haugen, and H.C. Benestad, "Using planning poker for combining expert estimates in software projects," *Journal of Systems and Software*, Vol. 81, No. 12, 2008, pp. 2106–2117.

[14] V. Campos, B. Jou, and X. Giro-i Nieto, "From pixels to sentiment: Fine-tuning CNNs for visual sentiment prediction," *Image and Vision Computing*, Vol. 65, 2017, pp. 15–22.

[15] K. Marasek et al., "Deep belief neural networks and bidirectional long-short term memory hybrid for speech recognition," *Archives of Acoustics*, Vol. 40, No. 2, 2015, pp. 191–195.

[16] K.S. Tai, R. Socher, and C.D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[17] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, Vol. 61, 2015, pp. 85–117.

[18] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, Vol. 25, 2012.

[19] K.I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, Vol. 6, No. 6, 1993, pp. 801–806.

[20] Y. Chen, "Convolutional neural network for sentence classification," Master's thesis, University of Waterloo, 2015.
[21] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola et al., "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.
[22] M. Jørgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, Vol. 70, No. 1–2, 2004, pp. 37–60.
[23] M. Jørgensen and T.M. Gruschke, "The impact of lessons-learned sessions on effort estimation and uncertainty assessments," *IEEE Transactions on Software Engineering*, Vol. 35, No. 3, 2009, pp. 368–383.
[24] B. Boehm, *Software cost estimation with COCOMO II*. New Jersey: Prentice-Hall, 2000.
[25] P. Sentas, L. Angelis, and I. Stamelos, "Multinomial logistic regression applied on software productivity prediction," in *9th Panhellenic Conference in Informatics*, 2003, pp. 1–12.
[26] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris, "Software productivity and effort prediction with ordinal regression," *Information and Software Technology*, Vol. 47, No. 1, 2005, pp. 17–29.
[27] S. Kanmani, J. Kathiravan, S.S. Kumar, and M. Shanmugam, "Neural network based effort estimation using class points for OO systems," in *International Conference on Computing: Theory and Applications (ICCTA'07)*. IEEE, 2007, pp. 261–266.
[28] A. Panda, S.M. Satapathy, and S.K. Rath, "Empirical validation of neural network models for agile software effort estimation based on story points," *Procedia Computer Science*, Vol. 57, 2015, pp. 772–781.
[29] S. Kanmani, J. Kathiravan, S.S. Kumar, and M. Shanmugam, "Class point based effort estimation of oo systems using fuzzy subtractive clustering and artificial neural networks," in *Proceedings of the 1st India Software Engineering Conference*, 2008, pp. 141–142.
[30] S. Bibi, I. Stamelos, and L. Angelis, "Software cost prediction with predefined interval estimates," in *Proceedings of Software Measurement European Forum*, Vol. 4, 2004, pp. 237–246.
[31] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering*, Vol. 23, No. 11, 1997, pp. 736–743.
[32] L. Angelis and I. Stamelos, "A simulation tool for efficient analogy based cost estimation," *Empirical Software Engineering*, Vol. 5, No. 1, 2000, pp. 35–68.
[33] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in *38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 619–630.
[34] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, 2006, pp. 33–53.
[35] E. Kocaguneli, T. Menzies, and J.W. Keung, "On the value of ensemble effort estimation," *IEEE Transactions on Software Engineering*, Vol. 38, No. 6, 2011, pp. 1403–1416.
[36] F. Collopy, "Difficulty and complexity as factors in software effort estimation," *International Journal of Forecasting*, Vol. 23, No. 3, 2007, pp. 469–471.
[37] E. Kocaguneli, T. Menzies, and J.W. Keung, "On the value of ensemble effort estimation," *IEEE Transactions on Software Engineering*, Vol. 6, No. 38, 2012, pp. 1403–1416.
[38] R. Valerdi, "Convergence of expert opinion via the wideband Delphi method," in *21st Annual International Symposium of the International Council on Systems Engineering, INCOSE*, Vol. 2011, 2011.
[39] S. Chulani, B. Boehm, and B. Steece, "Bayesian analysis of empirical software engineering cost models," *IEEE Transactions on Software Engineering*, Vol. 25, No. 4, 1999, pp. 573–583.
[40] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.
[41] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2016, pp. 1–10.
[42] C. Commeyne, A. Abran, and R. Djouab, "Effort estimation with story points and cosmic function points – An industry case study," *Software Measurement News*, Vol. 21, No. 1, 2016, pp. 25–36.

[43] G. Poels, "Definition and validation of a COSMIC-FFP functional size measure for object-oriented systems," in *Proc. 7th Int. ECOOP Workshop Quantitative Approaches OO Software Eng. Darmstadt*, 2003.
[44] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, and G. Succi, "Effort prediction in iterative software development processes – Incremental versus global prediction models," in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. IEEE, 2007, pp. 344–353.
[45] P. Hearty, N. Fenton, D. Marquez, and M. Neil, "Predicting project velocity in XP using a learning dynamic bayesian network model," *IEEE Transactions on Software Engineering*, Vol. 35, No. 1, 2008, pp. 124–137.
[46] M. Perkusich, H.O. De Almeida, and A. Perkusich, "A model to detect problems on scrum-based software development projects," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1037–1042.
[47] M. Choetkiertikul, H.K. Dam, T. Tran, T. Pham, A. Ghose et al., "A deep learning model for estimating story points," *IEEE Transactions on Software Engineering*, Vol. 45, No. 7, 2018, pp. 637–656.
[48] E. Giger, M. Pinzger, and H. Gall, "Predicting the fix time of bugs," in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 52–56.
[49] L.D. Panjer, "Predicting eclipse bug lifetimes," in *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*. IEEE, 2007, pp. 29–29.
[50] P. Bhattacharya and I. Neamtiu, "Bug-fix time prediction models: Can we do better?" in *Proceedings of the 8th Working Conference on Mining Software Repositories*, 2011, pp. 207–210.
[51] P. Hooimeijer and W. Weimer, "Modeling bug report quality," in *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*, 2007, pp. 34–43.
[52] E.M.D.B. Fávero, D. Casanova, and A.R. Pimentel, "SE3M: A model for software effort estimation using pre-trained embedding models," *Information and Software Technology*, Vol. 147, 2022, p. 106886.
[53] P. Liu, Y. Liu, X. Hou, Q. Li, and Z. Zhu, "A text clustering algorithm based on find of density peaks," in *7th International Conference on Information Technology in Medicine and Education (ITME)*. IEEE, 2015, pp. 348–352.
[54] J. Pennington, R. Socher, and C.D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.
[55] W. Guohua and G. Yutian, "Using density peaks sentence clustering for update summary generation," in *Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2016, pp. 1–5.
[56] J. Pennington, R. Socher, and C.D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.
[57] K. Cho, A. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *IEEE Transactions on Multimedia*, Vol. 17, No. 11, 2015, pp. 1875–1886.
[58] E. Loper and S. Bird, "Nltk: The natural language toolkit," *arXiv preprint cs/0205028*, 2002.
[59] B.A. Kitchenham, L.M. Pickard, S.G. MacDonell, and M.J. Shepperd, "What accuracy statistics really measure," *IEE Proceedings – Software*, Vol. 148, No. 3, 2001, pp. 81–85.
[60] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Transactions on Software Engineering*, Vol. 29, No. 11, 2003, pp. 985–995.
[61] M. Korte and D. Port, "Confidence in software cost estimation results based on MMRE and PRED," in *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering*, 2008, pp. 63–70.

[62] D. Port and M. Korte, "Comparative studies of the model evaluation criterions mmre and pred in software cost estimation research," in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2008, pp. 51–60.
[63] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in *38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 619–630.
[64] T. Menzies, E. Kocaguneli, B. Turhan, L. Minku, and F. Peters, *Sharing data and models in software engineering*. Morgan Kaufmann, 2014.
[65] K. Muller, "Statistical power analysis for the behavioral sciences," 1989.
[66] J. Cohen, *Statistical power analysis for the behavioral sciences*. Routledge, 2013.
[67] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[68] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.
[69] A.M. Dai and Q.V. Le, "Semi-supervised sequence learning," *Advances in Neural Information Processing Systems*, Vol. 28, 2015.
[70] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian et al., "GPT understands, too," *arXiv preprint arXiv:2103.10385*, 2021.
[71] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R.R. Salakhutdinov et al., "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in Neural Information Processing Systems*, Vol. 32, 2019.